Tree Cover and Demographics

Kelly Winkelmeyer and Candace Craven

Extract

* Our original data sources were csv files from denvergov.org.
* Read csv files into Jupyter notebooks with Pandas pd.read\_csv.

Transform

* Cleaning process:
  + Demographics dataframe:
    - First loaded demographics and counted, and listed columns to determine which columns to drop.
    - Used .drop with .iloc to drop unnecessary columns.
    - Used df.count to ensure each column had the same number of rows.
    - Renamed columns for easier reading.
    - Verified that all neighborhoods were unique by using df[‘column’].unique.
    - The columns that we kept, were all numeric values, which may have included zero values.
  + Tree dataframe:
    - First looked for duplicates using the site id, using df.drop\_duplicates.
    - Counted and listed columns to view columns and determine which ones to drop.
    - Renamed columns to be more reader friendly.
    - Used df.count to ensure each column had the same number of rows.
    - Verified that all neighborhoods were unique by using df[‘column’].unique.
* Merging:
  + After confirming the neighborhood column in each dataframe were compatible, the dataframes were merged on the neighborhood column.
  + Set the index to be the neighborhood id.
  + Created a table class in Python for the merged table.

Loading

* We connected to postgres using psycopg2.
* Then we created the database and then we connected to the database using create\_engine.
* Using the connection engine, the declarative base table is able to be created in postgres.
* The table data is then pushed to postgres using .tosql.
* To confirm upload to postgres, pd.read\_sql\_query was used to print table.
* Tables were manipulated in Pandas by preference which allowed only one table to be uploaded to the database.